11.0 Introduction

ForTran is supposed as the oldest high-level language. ForTran stands of **For**mula **Tran**slator. Fortran was developed in several stages and declared as ForTran77 in 1977 after incorporating essential features of a programming language such as file handling. In the subsequent text we will discuss about ForTran77. Like C, ForTran77 also uses its compiler to convert source code to object code. All the programs in this text are tested using Force 2.0 ForTran compiler.

Recently, ForTran is included in the curriculum of TU, IOE, BE (all programs) as topic of syllabus of COMPUTER PROGRAMMIG. The weight of this topic is 12 out of 80 marks and the allocated time duration is 8 hours. According to the syllabus, we should study about C programming language before studying ForTran. Concept of writing algorithm, drawing flowchart, flowcharting symbols and fundamental of programming is independent of any languages. It means, we are already familiar to those common topics. Therefore, it this text we mainly focus on the syntactic structure of ForTran programming language. Before proceeding, let us compare a basic example of a ForTran program to the same purpose C program. We can write ForTran syntax either in all upper case or all in lower case or in both.

Example 1: A program to calculate the simple interest when p, t and r are given in both ForTran and C. Compare these and notice the differences in program structure, syntactic form and case sensitiveness.

ForTran program	C program	Comment
	void main()	
	{	
rEal P,t,r,i	float p,t,r,i;	variable declaration
WRITE(*,*),'Enter value of p:'	Printf(“Enter value of p:”);	formatted output statement
read(*,*)p	scanf(“%f”,&p);	formatted input statement
Write(*,*)'Enter value of t:'	Printf(“Enter value of t:”);	formatted output statement
read(*,*),T	scanf(“%f”,&t);	formatted input statement
wRite(*,*)'Enter value of r:'	Printf(“Enter value of r:”);	formatted output statement
read(*,*)r	scanf(“%f”,&r);	formatted input statement
I=(p*T*r)/100	i=(p*t*r)/100	operators, operands and expression
writE(*,*)'The interest is:',i	Printf(“The interest is %f:”, i);	formatted output statement
END	}	end of program

Question: Write a program

a. to find the volume of a room when height, width and length is given.
b. to find the area and circumference of a circle when radius is given.
c. to find surface area and volume of a sphere when radius of the sphere is given.
d. to find value of $f(x)=x^2+3$ if value of x is given.
e. to convert the given Centigrade measure to Fahrenheit using relation F=1.8C+32.
f. to compute equivalent resistance of two resistors R_1 and R_2 when they are connected in series and parallel connection.
g. to read two end points of a line, compute their mid point and display it.

Note: see topic11.4 to know how to run a ForTran program using Force2.0 compiler.

11.1 Character set

The character set includes letters form A to Z or a to z. Similarly digits from 0 to 9 and symbols + - / * . , ‘ _ $ ( ) .

11.2 Constants, variables and data types

Constants

Fortran constants can be classified as:

A. Numeric constants
    a. Integer constants
    b. Real constants
        - Fractional form
        - Exponential form

B. Character constants

Integers constants

Integer numbers written without decimal point are called fixed-point constants or integer constants. These are also called whole number having no fractional part. Integers are formed by digits 0,1,2, 3,4, 5, 6,7,8,9 and + or - symbol. The sign symbol occurs at the leftmost position of the number. No sign symbol means positive integer. –445, 234, 045,+0,-0,0,2,12345 are some valid examples of integer. On the other hand, 12.3, **-2,45**, 123+7, 123-, 1.0 are not valid for integer. The maximum length of an integer depends upon the word length of the computer system. For 16 bits computer the range is -2^{15} to 2^{15}-1.

Real constants

If a numbers has a fractional part is called real or floating point constant. Such numbers are written with one decimal point. For example 4⅛ is a real number because it has 4 and one eighths i,e 4.125. The real constants can be written in either fractional or exponential form.

Fractional form

In this form, real numbers are written using only one decimal point, digits 0,1,2, 3,4, 5, 6,7,8,9 and + or - symbol. The sign symbol occurs at the leftmost position of the number. No sign symbol means positive number. 123.4, 345.8976, -0.4546, 0.0, -200.0 are some example of this form. Similarly,100. is also valid real constant because decimal point is at the end of the number. Conversely, 0, -(2.5), 123, 10/45,1.23+0.2, 45,89.67, 0.123-, 34.65.7 etc are some invalid real constants.

Exponential form

Sometimes, we may need to use too large and too small numbers in scientific computations. For example inter planet distance and weight of electron etc. For instance, 1230000000000 is a large number and it can be written as 0.123×10^{13}. Here 0.123 is called **mantissa** and 13 is called the **exponent**. In ForTran it is written as 0.123E13. Letter E is used between mantissa and exponent. From this discussion, we can generalize the exponential form as **mantissa** E **exponent.** The **mantissa** must be a valid real constant in fractional form and exponent must always be an integer. The exponent can be preceded by a + or – sign. No sign means positive number. The range of real constants depends on the word length of the underlying computer system. Some other examples are listed in the following table.

Number	Normal exponential form	Fortran exponential form	Mantissa	Exponent
-1230000000000	-0.123×10^{13}	-0.123E13	-0.123	13
0.0000000000123	0.123×10^{-13}	0.123E-13	0.123	-13
-0.0000000000123	-0.123×10^{-13}	-0.123E-13	-0.123	-13

Example 2: This program reads three sets of too small and too large numbers and displays them to illustrate the concept of range of real constants.

```
real x,y                                    ! real type variable declaration like float x,y in C.
write(*,*)'Enter values of x and y:'        ! like printf("Enter values of x and y"); in C.
read(*,*)x,y                                ! scanf("%f%f", &x,&y); in C
write(*,*)'Value stored at x:',x,'Value stored at y:',y
                                            ! like printf("Value stored at x value stored at y", x,y); in C.
end                                         ! end of a ForTran program
```

! is a symbol used to comment in ForTran program like /* */ in C.

Output:

Run1:

Enter values of x and y:
-654664113687974643215662132 06.987898 5654651635867698416316354654232 6.23546
Value stored at x: -6.546641E+28 Value stored at y: 5.654652E+31

Run1:

Enter values of x and y:
-3.4E-38 3.4E38
Value stored at x: -3.4E-38 Value stored at y: 3.4E+38

Run2:

Enter values of x and y:
-12345678910111213141516171819 20 1234567891011121314151617181192 0
Value stored at x: -1.2345679E+30 Value stored at y: 1.2345679E+31

Character constants

These constants are similar to the string constants in C. But in case of ForTran these constants are enclosed by single quotes (‘). For example, ‘Patan’, ‘Pulchowk campus’, ‘total amount = ’ ,’c’are some valid examples of string constant. Single quotes can not be the part of a string. If required we can use “ instead e.g. Gita”s instead of Gita’s. Following examples example illustrate reading and printing string constants of different length.

Example 3: Write a program to read and display character constants of different length.

```
character x*130,y,z*20          !character variable declaration.
write(*,*)'Enter characters for character constants x,y and x:'
read(*,*) x,y,z                 !reading three character variable
write(*,*)'Content of x:',x     !printing content of x
write(*,*)'Content of y:',y     !printing content of y
write(*,*)'Content of z:',z     !printing content of z
end
```

Output:

Enter characters for character constants x,y and z:

wqeqeqeqwewewqeqweqwewqewqeqweqwweqweewqeqweqwewqeqweqwewqewqeqweqwewqeqweqwe
qweqeqweqwewqeqeqweqweqweqweqweqweqwewqewqeqwewqewqeqewerewrwrwerwereer
ert45
er4535dgdgerttrt5677yy09-=\

Content of x:wqeqeqeqwewewqeqweqwewqewqeqweqwweqweewqeqweqwewqeqweqwewqewqeqweqwewqeqweqwewqeqeq
Content of y:e
Content of z:er4535dgdgerttrt5677

Fortran Verbs: what is the role of keywords in C? There are some words in ForTran having the similar role are called ForTran varbs. program, real, integer, character, format, write, read, print, if, then, else, elseif, endif, do, goto, continue, while, data, stop, end, call, common, complex, dimension, endfile, logical, subroutine, entry, external, function, intrinsic, double precision are some examples of such words.

Variables

The concept of variable is already studied in C. Actually variables are the name of memory location where the data are stored for computing. Depending on the type of data they hold these are classified as integer, real and character variables in ForTran. The general form of integer variable declaration is as

integer list of variables (separated by commas)

for example, the statement **integer i, length, amount** declares i, length and amount as integer variables. These can be declared in separate lines. Similarly, the general form of real variable declaration is as

real list of variables (separated by commas)

for instance, the statement **real i, length, marks** declares i, length and amount as real variables. These can also be declared in different lines as:

real i,
real length, marks

Likewise, the general form of character variable declaration is

character list of character variables (separated by commas).

For example, the statement **character x*130,y,z*20** declares x as a character variable of length at most 130, y of length 1 because no length is specified there and z of length 20.

Important: The variable name starting with letters **i,j,k,l,m** or **n** are considered to be integer variables and others are real variables. Therefore, it is not mandatory to declare variables of type integer and real if their name is started with the defined alphabets.

There is another way of variable declaration i,e, implicit type declaration. The general form of this type is

implicit type (…), type (…-…)

for example **implicit real(j)** indicates that variables starting with j are real. Now there are only i, k, l, m, or n for integers for defined condition. Some other examples are listed in the following table.

Declaration	**Comment**
implicit integer(b)	Variables starting with b are considered as integer. Therefore, the letter those makes variables integer are b,i,j,k,l,m,n
implicit integer(p,u-z), real (m)	This declares that variable names starting with p,u,v,w,x,y,z or i,j,k,l,n are integers
implicit character *50(r-t)	It means variable name starting with r, s, t are character variables having capacity at most 50 characters.

Naming variable: The first character of any variable name must be an alphabet. ForTran verbs can not be used for variable. Similarly any especial characters are not allowed in variable names. For example, 7sum, goto, r-e are some invalid variable names. It is better to give meaningful variable name. For instance, if we are going to calculate area of a circle, we can write area, AREA and any other meaningful name for the variable. It makes easier to understand the program.

Data types

Prior to using a particular variable, we have to specify its data type to tell computer to allocate a sufficient amount of space for storage of this data. The variables are declared at the beginning of a program. We have just studied about integer, real and character data types. These are the main data types used in ForTran programming to cover the objectives of our course. There are other data types that are listed in the following table.

Data type name	**Size**	**Description**
logical	1 bit	stores either logical 0 (.false.) or logical 1 (.true.)
byte	1 byte	integral numbers from -128 to 127
integer	4 bytes	integral numbers from -2^{31} to $2^{31}-1$
real	4 bytes	real numbers approx. between 1.2×10^{-38} and 3.4×10^{38}
double precision	8 bytes	real numbers approx. between 2.2×10^{-308} and 1.8×10^{308}
complex	2 x 4 bytes	two real numbers, real and imaginary part
double complex	2 x 8 bytes	two double precision numbers, real and imaginary part
character	1 byte	one alphanumeric character
character*N	N bytes	N alphanumeric characters
Source: Roman GrÄoger,University of Pennsylvania, Department of Materials Science and Engineering, 3231 Walnut Street, Officece #215, Philadelphia, PA 19104, Revision 1.2 (September 27, 2004)		

11.3 Arithmetic operators, Library Functions

Arithmetic operators
The list of arithmetic operators and their operations are listed in the following table.

Symbol	Operation	arithmetic expression	Meaning
+ (plus)	addition	x+y	x and y are added
- (minus)	subtraction	x-y	y is subtracted from x
* (asterisk)	multiplication	x*y	x is multiplied with y
/ (slash)	division	x/y	x is divided by y
**(double asterisk)	exponentiation	x**y	x^y is evaluated or x is multiplied y times

The way of writing mathematical expressions in ForTran is different from the usual one. Some of those examples are shown in the following table.

In general mathematics	In ForTran
$2ax^4+bx^3+cx^2+2dx/5$	2*a*x**4+b*x**3+c*x**2+(2*d*x/5)
$A^{10/15}$	A**(10.0/15.0)
$\frac{a+b}{c+d}$	(a+b)/(c+d)
$(a/b)^p+30$ and $(a/b)^{(p+30)}$	(a/b)**p+30 and (a/b)**(p+30)

The concept of integer, real and mixed mode arithmetic is similar to that as we studied in C.

Assignment operator (=)
The purpose of this operator is to assign the result generated by the arithmetic expression on its right hand side to the single variable on its left hand side. The general form is

variable = expression The expression may be a constant or a variable.

For example consider the expression **x=p**3+2*p+6**. Here if the value of p is 2 then the result of expression p**3+2*q+6 (18) is stored at variable x. Any expressions created using assignment operators are called assignment expressions. There are four cases for integer and real variables in such expressions. These are listed in the following table.

Cases	Comment
integer variable=integer expression	The integer expression gives the integer result, which is assigned to integer variable.
integer variable=real expression	The real expression gives the real value, then the fractional part is truncated and only the integer part is assigned to integer variable.
real variable= real expression	The real expression gives the real result, which is assigned to real variable.
real variable= integer expression	The integer expression results in an integer value which is converted into real and assigned to real variable. For example, if 5 is generated from the expression 5.0 is assigned to the real variable.

Library or built-in or intrinsic functions

Main library functions available in ForTran are listed in the following table. These functions are directly useful in our programs. The compiler itself evaluates these functions. For example, if lox10(x) is a function , then x is its argument.

Function	Meaning	Argument type	Return type
abs(x)	x	Real or integer	Real or integer
exp(x)	e^x	Real	real
sqrt(x)	$\sqrt{x}$	Real	real
log(x)	$\log_e x$	Real (x>0)	real
log1o(x)	$\log_{10} x$	Real (x>0)	real
sin(x)	sin x	Real, in radian measure	real
cos(x)	cos x	Real, in radian measure	real
tan(x)	tan x	Real, in radian measure	real
asin(x)	$\sin^{-1} x$	Real,$-1 \le x \le 1$	Real, in radian measure
acos(x)	$\cos^{-1} x$	Real,$-1 \le x \le 1$	Real, in radian measure
atan(x)	$\tan^{-1} x$	Real	Real, in radian measure
sinh(x)	sinh (x)	Real	real
cosh(x)	cosh (x)	Real	real
tanh(x)	tanh (x)	Real	real
int(x)	*	Real	integer
nint(x)	**	Real	integer
max(x1,x2,...)	**	Real or integer	Real or integer
	* converts the argument to an integer value		
	** rounds x to the nearest integer value		
	*** returns the maximum value of x1, x2, ...		

Example 4: This example illustrates the usage of some library functions.

```
real x,y,z,m,r,ni   ! variable declaration
Write(*,*)'Enter values of x,y and z:'
read(*,*) x,y,z
m=max(x,y,z)
r=int(y)
ni=nint(z)
write(*,*)'Content of m:',m,'content of r:',r,'content of ni:' ,ni
write(*,*)'absloute value of x:',abs(x)
write(*,*)'Square root of y+z:',sqrt(y+z)
end
```

Output:

```
Enter values of x,y and z:
-5.5  30.7   5.6
Content of m: 30.7 content of r: 30. content of ni: 6.
 absloute value of x: 5.5
 Square root of y+z: 6.024948
```

Question: Write programs to show the usage of each function listed in the above table.

11.4 Structure of a Fortran Program
The main parts of a ForTran programs are

Program name
declaration of variables
initial definition of variables
main program body
end

Program name: The first line of the programs generally gives the program's name. For example, **program factorial** indicates that the following program is about computing. Here program is a ForTran verb but factorial is identifier. We can write whatever we like. In fact, this part of a ForTran program is optional. But it is good practice to name every program.

Declaration of variables: In this we must declare all the variables that will be used to store data, which will be used in computation in executable part. We have already discussed about ForTran data types and constants. If there is no variable in our program, we need to declare any variable unnecessarily.

Initial definition of variables (Initialization): After the declaration of variables, our computer knows how much space is to be allocated for each variable, but the actual value of this variable is not known. Although some ForTran77 compilers implicitly assign each declared variable a zero value or an empty string (for character types), it is strongly recommended to initialize variables if any variable needs to set some initial value.

Main program body: The main program body contains all executable ForTran statements that will carry out the task we wish to perform. These statements are executed in top to bottom order. In main program, a number of sub programs can be called to make main program more efficient. This mechanism is similar to calling user-defined functions in C.

End: End is the last statement of the ForTran program. It indicates the end of the program. Every ForTran program must be terminated by the **end** statement.

Example 5: A sample ForTran program. The first line in this program is the comment line. Such lines can be placed anywhere in the program. For commenting, we can use either C,* or !. To use C or * for commenting it must be placed at the first column of the ForTran coding sheet but symbol **!** can be placed anywhere except inside single quotes.

```
C   sample ForTran Program
    program sum                          ! here name of the program is sum. This line is optional
    real a,b,c,d                    !variable declaration section
    d=5                             !variable initialization section
    write(*,*)'Enter value of a and b:'  !like printf function in C
    read(*,*)a,b                         !like scntf function in C
    c=a*b-d
    write(*,*)'Value stored at c is:',c      !like printf function in C
    end          !end of the program. Actual program is between program name statement and
the  end statement.
```

Output:
Enter value of a and b:
4.5
5.5
Value stored at c is: 19.75

ForTran Coding sheet: It is a sheet of paper used to write ForTran program.

- It has **80** columns.
- Column **1** is used for writing C or * to indicate the the line as comment line.
- Columns **1 to 5** are used for giving line numbers for other lines of program. The line number is a positive integer with at most five digits. Every statement need not be given the line number.
- ForTran statements are written from column **7 to 72**.
- The computer does not read columns **73 to 80**. These are for comment of the programmer if needed
- **Sixth** column is used for continuation i,e, if a statement does not fit in columns 7 to 72, the remaining part must be written in the next line. For continuation same character must be placed at the last column (72) of current line and 6th column of the next line.

This discussion is illustrated in the coding sheet as shown in figure below by writing a sample program on it.

```
1 2 3 4 5 6 7 8 9 . . . . . 80
C           s a m p l e   F o R T r a n   P r o g r a m
            p r o g r a m   s U m
            r e a l   a , b , C , d
            d = 5
            w r i t e ( * , * ) ' E N t e r   v a l u e   o f   a   a n d   b : '
            r e a d ( * , * ) A , b
            c = a* b - d
            w r i t e ( * , * ) ' v a l u   s t o r e d   a t   c   i s   : '
            s t o p
            e n d
```

Figure: Illustration of ForTran Coding sheet and a sample program on it.

We can see the similar configuration on the editor of Force2.0 ForTran compiler. To run the program simply the type the code as discussed above and click on run➔compile. If errors occurs, debug them else click on run➔run. (for **stop** statement see page 16, example 13)

11.5 Formatted and Unformatted Input Output statements

Formatted Input Output

As in C, ForTran has some rules for formatted input and output while inputting and outputting data. The specification of type and its size is called format specification. For formatting, there is a non executable statement **format**. Its general form is

N format(f_1,f_2,f_3….f_n)

Where,

- **N** is the statement number and must be for each format statement. Format statements can be placed anywhere in the program.
- **f_1,f_2,f_3….f_n** are the format specifications and these must be separated by commas and enclosed within parenthesis.

The formats in ForTran for different purposes are I,X,F,E,/,A,T and quote ‘’

I format: I format is for integer. Its general form is Iw. Where w is the width of the integer data. For example –13 has width 3, 1245 has width 4 and +5 has width 2. The format statement for these three numbers is **format(I3,I4,I2).** If this format is used for reading data, value for the first integer must be read from columns 1 to 3, for the second from columns 4 to 7 and for third from columns 8 to 9 of the screen. If it is used for printing data, value of the first integer must be printed on columns 1 to 3, second on columns 4 to 7 and of third on column 8 to 9 of the output screen. This is illustrated in the following figure.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
-	1	3	1	2	4	5	+	5											

If **n** integers have same width **w**, the format statement can be written as **format(n*Iw).** While outputting data, one character may be lost because of carriage control. If so, the first character of the output data must be a blank space.

X format: x format is used to skip column(s) while printing data. It can be used only for printing data on the screen. Its general form is **nX**. Where n is the number of columns to be skipped. If we want to skip two columns between –13 and 1245 & 1245 and +5, the format specification becomes **format(I3,2X,I4,2X,I2).** Then the output will be as

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
-	1	3			1	2	4	5			+	5							

Example 6: This program illustrates format(I3,I4,I2) to read three integers and format(I3,I4,I2)& format(I3,2X,I4,2X,I2) to print the integers.

```
      integer f,s,t
      write(*,*)'Enrer value of f,s,t:'
      read(*,1)f,s,t                    !reading data in format 1
1     format(I3,I4,I2)                  !format statement 1
      write(*,*)'Data according to format1:'
      write(*,1)f,s,t                   !writing or printing data in format 1
      write(*,*)'Data according to format2:'
      write(*,2)f,s,t                   !writing or printing data in format 2
2     format(i3,2x,I4,2x,I2)            !format  statement 2
      end
```

Output:

```
Enrer value of f,s,t:
-131245+5
 Data according to format1:
-131245 5                    ! + symbol is not displayed
 Data according to format1:
-13  1245   5                !  + symbol is not displayed
```

Here 1 and 2 are format statement number. It is also called the statement label. Any integer from 1 to 99999 can be used for this purpose. But we cannot repeat the same number for different statements in a program.

read statement: this statement works like **scanf()** function in C. This means the **read** statement is used for transferring the data from input device to the main memory of the computer. Its general form is

read(h, N) a1, a2, a3….

Here, **h** is an integer and it indicates the hardware number from which the data to be read. In a computer system there may be a number of input devices such as card reader, magnetic tap, and keyboard. The devices are given unique numbers. For example, 1 for card reader, 2 for magnetic tape and 3 for keyboard. Keyboard is the most common input device. Therefore, it is considered as the default input device. The * in the position of hardware number means default input device i,e, keyboard. The variables listed in the **read** statement must have one to one correspondence with the format specification in the format statement.

Question: what does * in position of hardware number mean in read statement **read(*,1)f,s,t in above example 6**?

Similarly, **N** is the format statement number according to which values of variables **a1,a2,a3…**to be printed. In format statement **read(*,1)f,s,t ,** 1 in place of N means read data according to format statement number 1. Like in hardware number, what happens when * is used in place of format statement number? The answer is easy that the read statement reads data in default format. From the discussion, we can say that the statement **read(*,*)f,s,t** , reads values of f,s and t from the default hardware in the default format.

write statement: this statement works like **printf()** function in C. This means the **write** statement is used for outputting the result. Its general form is

write(h, N) a1, a2, a3….

Here, **h** is an integer and it indicates the hardware number of output device. In a computer system there may be a number of output devices such as printer, monitor etc. The devices are given unique numbers. For example, 6 for printer and 3 for VDU(Dideo Display Unit). VDU is the most common output device. Therefore it is considered as the default output device. The * in the position of hard ware number means default output device i,e, VDU.The variables listed in the **write** statement must have one to one correspondence with the format specification in the format statement.

Question: what does * in position of hardware number mean in **write** statement **write(*,1)f,s,t** in above example 6?

Similarly, **N** is the format statement number according to which values of variables **a1,a2,a3…**will be printed. In format statement **write(*,1)f,s,t ,** 1 in place of N means to print data according to the format statement number 1. Like in place of hardware number, what happens when * is used in place of format statement number? The answer is easy that the **write** statement prints data in default format. From the discussion, we can say that the statement **write(*,*)f,s,t** , prints value of f,s and t to the default hardware in the default format.

F format: This format is used for real data expressed in decimal form. The general form of this format is **Fw.d.** Where **w** is the total width and **d** is decimal width of the number. For example, -12.345 has total width(w) 7 and decimal width(d) 3. The format specification for this number is F7.3. Similarly, format specification for 1.2346 is F6.4. The format statement to read these two read data is **format(F7.3,F6.4)** and to print these two data with three spaces between them is **format(F7.3,3x,F6.4)**. These are shown in the figure below.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
-	1	2	.	3	4	5	1	.	2	3	4	6							

Figure: illustration of **format(F7.3,F6.4)** for reading data

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
-	1	2	.	3	4	5				1	.	2	3	4	6				

Figure: illustration of **format(F7.3,3x,F6.4)** for printing data

Example 7: This example illustrates the concept of reading and writing real data using F format.

```
      real f,s
      write(*,*)'Enter values of f and s:'
      read(*,1)f,s
1     format(F7.3,F6.4)
      write(*,*)'Values stored at f and s are:'
      write(*,2)f,s
2     format(F7.3,3x,F6.4)
      end
```

Output:

```
Enter values of f and s:
-12.3451.2346
 Values stored at f and s are:
-12.345   1.2346
```

E format: It used for real data in exponential form. Its general form is Ew.d. Where **w** is the total width and **d** is the decimal width of the mantissa. For example, **w** for -0.12E-12 is 9 and d is 2 and w for 1.23E12 is 7 and d is 2. Format specifications for these numbers are E9.2 and E7.2 respectively. The format specification statement for reading these two data is **format(E9.2,E7.2)** and for printing with 2 spaces at the beginning and 2 spaces between them is **format(1x,E9.2, 2x, E7.2).** These two formats are illustrated in the following figures.

1	2	3	4	5	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
-	0	.1	2	E	-	1	2	1	.	2	3	E	1	2					

Figure: illustration of **format(E9.2,E7.2)** for reading data

1	2	3	4	5	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
		-	0	.1	2	E	-	1	2			1	.	2	3	E	1	2	

Figure: illustration of **format(1x, E9.2, 2x, E7.2)** for printing data

Example 8: This example illustrates the concept of reading and writing real data using E format.

```
      real f,s
      write(*,*)'Enter values of f and s:'
      read(*,1)f,s
1     format(E9.2,E7.2)
      write(*,*)'Values stored at f and s are:'
      write(*,2)f,s
2     format(2x,E9.2,2x,E7.2)
      end
```

Output:

```
Enter values of f and s:
-0.12E-121.23E12
Values stored at f and s are:
 -0.12E-12  .12E+13
```

/(slash) format: This is specially used for printing data. It is used to skip to the next line like \n in C.

Example 9: This example illustrates the concept of writing data using slash and quote format

```
      read(*,1)i,j            !why I and j are not declared here?
1     format(I4,I3)
      write(*,2)i,j
2     format('value at i:',I4/'value at j:',I3) !use n slashes for skipping n lines e.g. // for skipping two lines
      end
```

Output:

```
1234567
value at i:1234
value at j:567
```

A Format: It is used for character data. The general form is **Aw**. Where **w** is the width. The w is optional here. If w is not specified then the size of variable declared in the **character** statement is considered as the width. For example,

Character x*12

X= 'Mt. Everest'

Output for **format(A)** and **format(A4)** are as illustrated in the following figure.

1	2	3	4	5	6	7	8	9	10	11
M	t	.	E	V	e	r	e	s	t	

Figure: illustration of **format(A)** for printing data

1	2	3	4	5	6	7	8
M	t	.	E				

Figure: illustration of **format(A4)** for printing data

T format: This format is used to give tab. Its general form is **Tn**. Where **n** stands for column from which the output must start. It is used only for output statements. For example, let us take a format statement,

format(F8.2,T12,I3)

First value will be printed in 8 columns and in F format and the second value will start from column 12.

Quote Format: This is similar to " " in C. whatever is in the single quote, will be printed as it is. For example, if we need to print NEPAL the format statement for this task is **format('NEPAL').**

Unformatted Input Output

When unformatted Input Output statements are used the computer itself decides the formats. The **read** and **write** are unformatted input output statements.

read statement: The general form of this statement is **read*, a1, a2, a3…..** Where a1, a2, a3….are the list of the variables. When this statement is executed, the cursor blinks on the screen waiting for getting data. The supplied data must match with the type of the corresponding variable. The character data must be enclosed in single quotes. If there are more than one input data, they must be separated by spaces.

print statement: The general form of this statement is **write*, I1, I2, I3….** Where I1, I2, I3…. is the list of items to be printed. The items may include variables, constants, arithmetic expressions and character constants. These will be printed continuously with blank spaces between two items.

Example 10: This example illustrates the concept of unformatted Input output using read and print statements.

```
character Nam*15
integer roll
real m1,m2,m3
print*,'Enter name:'
read*,nam
print*,'Enter roll:'
read*,roll
print*,'Enter marks in 3 subjects:'
read*,m1,m2,m3
print*,'Name:', nam, 'roll:', roll, 'average marks:', (m1+m2+m3)/3
end
```

Output:

```
Enter name:
Prasiddhi
Enter roll:
1234
Enter marks in 3 subjects:
60.5    80.5     70.5  !if we need to enter same value for three variables for example 70.5, we can type 3*70.5
Name:Prasiddhi      roll: 1234  average marks: 70.5
```

Question: Write a program to compute the return amount (A) given by the expression $A=P(i(1+i)^n/((1+i)^n-1))$ on investment of P amount of money for n numbers of year and at interest rate i.

11.6 Control structures: goto, Logical IF, Arithmetic IF, Do loops

goto

Unconditional goto

This statement is used to transfer the program control to any other statement unconditionally. Its general form is:

goto k

Where, **k** is the statement number where the control must be transferred without depending on any condition. The value of **k** must not be floating point number, zero or negative. It can be five digits at most. There must be one statement number for each statement. **goto k** can be written as **go tok**, **Go Tok**, **GOTO k** etc. But **g oto k**, **g o t ok**, **go t o k** etc are invalid.

Example 11: In this example the **goto** is used for backward jump. Here goto statement is executed at line 5 and control is transferred to second line of the program. This program is going on in a close path infinitely because there is no exit point from the structure. Actually, it is in never ending loop. Such happening must be avoided in our programs. Techniques to handle such situation be implemented using **if** structure within such close paths.

```
      integer x
1     write(*,*)'Enter value of variable x :'
      read(*,*),x
      write(*,*)'The value stored at variable x is:',x
      goto 1
      end
```

Output:

```
Enter value of variable x :
23
 The value stored at variable x is: 23
 Enter value of variable x :
45
 The value stored at variable x is: 45
 Enter value of variable x :
56
 The value stored at variable x is: 56
Enter value of variable x :
```

Example 12: This example shows the forward jump of the goto statement. Here, goto is executed as the second statement and control is transferred to fifth statement i,e third and fourth statements are skipped. This skipping causes not to read the value of x. Therefore, the fifth statement prints the garbage value as shown in the output of this example.

```
      integer x
      goto 20
      write(*,*)'Enter value of variable x :'
      read(*,*)x
20    write(*,*)'The value stored at variable x is:',x
      end
```

Output:
The value stored at variable x is: 2293556

Computed goto statement
The general form of this type of goto statement is:

goto (k_1,k_2,k_3,……k_n), i

Where **k_1,k_2,k_3,……k_n** are statement numbers. These must be enclosed within parenthesis. The statement number may be repeated and may be in any order. The variable **i** is the single integer variable. There must be a comma between closing parenthesis and the integer variable like **i** in the above general form. The program control goes
to statement **k_1**, if the value of **i** is equal to **1**
to statement **k_2**, if the value of **i** is equal to **2**
to statement **k_3**, if the value of **i** is equal to **3**
……
……
to statement **k_n**, if the value of **i** is equal to **n**
if the value of **i<1 or i>n**, the **goto** statement itself is ignored and control goes to the next line.

Example 13: This program illustrates the use of computed goto statement. In this program, if the entered value is 1 the control goes to statement 15 and to 20 if the entered number is 2. Which is shown in the output as two different runs of the program.

```
      integer day;
      write(*,*)'Enter day no:'
      read(*,*)day
      goto(15,20), day
 15   write(*,*)'Sunday'
      goto 23                ! this goto statement transfers the control to the stop statement.
                             ! otherwise control goes to immediate next statement.
 20   write(*,*)'Monday'
      goto 23
 23     stop
      end
```

Output:

Run1:	**Run2:**
Enter day no: 2 Monday	Enter day no: 1 **Sunday**

stop statement is used to stop the program execution. It can be placed anywhere in the program. There may be more than one stop statements in a program. Normally it is placed just before the end statement. But it is optional in ForTran 77. Similarly, end statement indicates the end of the program. There must not be more than one end statements in a program.

Question: Differentiate between stop and end statement.

Suppose a goto **statement goto(5,5,5,2,3,66), i** . In such situation, control goes to statement
5 if the value of i is 1,2 or 3
2 if the value of i is 4
3 if the value of i is 5
66 if the value of i is 6

Logical if
The logical if statement uses relational and logical expression to transfer program control as in the C programming language. These expressions generate either true or false result depending on the operands and operator in them. Let us see the relational and logical operators used in C and their equivalent form in ForTran in the **following table**.

Relational operators in C	**Equivalent symbol in ForTran**
less than(<)	.lt.
less than or equal to (<=)	.le.
greater than(>)	.gt.
greater than or equal to (>=)	.ge.
equal to (==)	.eq.
equal to (!=)	.ne.
Logical operators in C	**Equivalent symbol in ForTran**
and (&&)	.and.
or (||)	.or.
not (!)	.not.

For example if x=100 and y=20 then
Is **x .gt. y** ? Yes. It means **true**
Is **x.eq. y** ? No. It means **false**
Is **(x .gt. y) .and. (x.eq. y)? ➔true .and. false .** It results in **false.**
Is **(x .gt. y) .or. (x.eq. y)? ➔true .and**. **false .** It results in **true.**
Is **.not. ((x .gt. y) .or. (x.eq. y))?➔ .not.(true .and. false) .➔.not.** (**true) .** Therefore, the final result becomes **false.**

The general form of logical **if** statement is **if(condition) statement** .Where condition is a relational or logical expression that gives either true or false result and **statement** is an executable statement. This concept is illustrated in following example.

Example 14: Write a program to read two real numbers x and y, add five to x if it is grater than y, subtract five from it if it is less than or equal to y and display the result.

```
      real x,y
      write(*,*)'Enter value of x and y:'
      read(*,*)x,y
      if(x.gt.y) goto 1
      x=x-5
      goto 2
1     x=x+5
2     write(*,*)x
      end
```

Output: (two different runs)

Enter value of x and y: 30 20 35.	In this case x is greater than y i,e, 30>20. The result is true and the control goes to statement number 1. 5 is added to x and the result is displayed.
Enter value of x and y: 20 30 15.	In this case x is less than y i,e, 20<30. The result is false and the control goes to immediate next statement. 5 is subtracted from x and control is transferred to statement 2 to display the new value of x.

Example 15: Write a program to evaluate the following expression. This example also illustrates the concept of logical if statement.

$f(x) = ax^2+bx+5$ if $x<5$

$= bx^2+ax-15$ if $5 \leq x$

```
      real a,b,x,fx
      write(*,*)'Enter the values of a,b and x:'
      read(*,*)a,b,x
      if(x.lt.5) goto 1
      goto 2
1     fx=a*x**2+b*x+5
      write(*,*)'The value of f(x):',fx
      goto 10
2     fx=b*x**2+a*x-15
      write(*,*)'The value of f(y):',fx
      goto 10
10    stop
      end
```

Output:

```
Enter the values of a,b and x:
2 3 5
The value of f(x): 70.
```

Question: A stationary has a provision to give 10 % discount to the customers who purchase for more than Rs. 2500. Write a program to calculate the discount for a customer on his purchase.

Question: Write a program to read an positive integer and display the message whether it is odd or even.

Question: write a program to read secured marks of a student and display PASS if the marks is greater than 40% else display FAIL.

In the following example, **concept of loop**(as we studied in C) is implemented using **logical if** statement.

Example 16: Write a program to check whether a positive integer read from the keyboard is palindrome or not. A number is palindrome if its reverse is equal to the number itself.

```
      integer n,sum,r,x
      sum=0
      write(*,*)'Enter a positive integer:'
      read(*,*) n
      x=n
```

```
 2    if(x.eq.0)goto 1
            r=mod(x,10)
            sum=sum*10+r
           x=x/10
      goto 2
 1    if(sum.eq.n)then
          write(*,*) 'The number',n,'is a palindrome.'
      else
          write(*,*) 'The number',n,'is not a palindrome.'
      endif
      end
```

Output:

```
Enter a positive integer:
1234321
1234321
The number 1234321 is a palindrome.
```

Questions: Write a program to read a positive integer from the keyboard and

a. count number of digits in it.
b. count odd and even digits in it.
c. find the sum of digits in it.
d. check whether the number is Armstrong number or not and modify the program to list three digits Armstrong numbers.
e. count prime and composite digits in it.
f. add individual digits contained in it until the final sum is a single digit. For example, if entered number is 2175698 then 2+1+7+5+6+9+8=>38=>3+8=>11=>1+1=>2

Example 17: Write a program to evaluate the series sin(x)= $x-x^3/3!+x^5/5!-x^7/7!$……..

This program sums the terms until the absolute value of latest term becomes less then 0.000001. This program can be used to find sine of any angle. In this series value of x is radian but we are more familiar to degree. So, the user is asked to input value of angle in degree and it is converted to radian before computing. Here every second term can be computed by multiplying the first term by **$-x^2/i(i+1)$** i is a variable and its value is 2 for second term, 4 for third, 6 for fourth and so on. How every term is generated is as follows:

first term=x

second term = - first term* $x^2/i(i+1) = -x* x^2/2(2+1) = -x^3/2.3 = -x^3/3!$ (when i=2**)**

third term = - second term* $x^2/i(i+1) = -(-x^3/3!) * x^2/i(i+1) = x^5/3!.4.5 = x^5/5!$ (when i=4)

fourth term = -third term* $x^2/i(i+1) = -(x^5/5!)* x^2/i(i+1) = -x^7/5!.6.7 = -x^7/7!$ (when i=6) and so on.

```
      real deg,rad,sum,term,i
      sum=0
      i=0
      write(*,*)'Enter the value of angle in degree:'
      read(*,*) deg
      rad=deg*3.14/180 !to convert degree to radian but it is not mandatory
      term=rad
      write(*,*) 'Term:',term !to print first term
 3    if(abs(term).lt. 0.000001)goto 1
          i=i+2
          sum=sum+term
          term=-term*rad*rad/(i*(i+1))
```

```
            write(*,*) 'Term:',term  !to print term
        goto 3
1       write(*,*)'sin(',deg,')=',sum  !to print sum of terms
        end
```

Output:

```
Enter the value of angle in degree:
30
Term: 0.5233334
Term: -0.023888234
Term: 0.00032712286
Term: -0.0000021331357
Term: 8.114147E-9
sin( 30. )= 0.4997701
```

Question: Write a program to evaluate the series cos(x)= $1-x^2/2!+x^4/4!-x^6/6!+x^8/8!-$……..until the absolute value of the latest term becomes less than 0.000001. Here every second term can be computed by multiplying the first term by **$-x^2/i(i-1)$** i is a variable and its value is 2 for second term, 4 for third, 6 for fourth and so on.

Question: Write a program to evaluate the exponential series $e^x = 1+x+x^2/2!+x^3/3!+x^4/4!+x^5/5!$…… until the latest term becomes less than 0.000005 when x=1.

Arithmetic if statement

The general form of this statement is **if(expression) k1,k2,k3**. Where **expression** is any valid ForTran arithmetic expression. k1,k2 and k3 are the statement numbers. If the value of expression is negative control goes to the statement number k1 and to k2 if the its value is zero. Similarly, control goes to statement k3 if the expression generates a positive value. This concept is illustrated in the following **example.** It asks the user to enter the three coefficients of the quadratic equation $ax^2+bx+c=0$. As we know, roots are

imaginary if $b^2-4ac<0$, which are $(-b\pm i\sqrt{|(b^2-4ac)|})/2a$.
Where $-b/2a$ is real part and $\sqrt{|(b^2-4ac)|}/2a$ is imaginary part
real and equal if $b^2-4ac=0$, which are $-b/2*a$ and $-b/2*a$
and real and unequal if $b^2-4ac>0$, which are $(-b\pm\sqrt{(b^2-4ac)})/2a$

Example 18: Write a program to read an integer from the user and display appropriate message whether it is positive, zero or negative using the concept of arithmetic if statement.

```
        real x
        read(*,*)x
        if(x)1,2,3
1       write(*,*) '-ve'
        goto 100
2       write(*,*) '0'
        goto 100
3       write(*,*) '+ve'
        goto 100
100     stop
        end
```

Output:

Run1:	***Run2:***	***Run3:***
Enter value of i: -1000 -ve	Enter value of i: 78 +ve	Enter value of i: 0 0

Example 19: This program to calculate all roots of a quadratic equation. Here, if b^2-4ac<0, control goes to statement 5, to 4 if b^2-4ac=0 and to 10 if b^2-4ac>0

```
      real a,b,c,r1,r2,d,rp,ip
      write(*,*)'Enter the values of a,b and c:'
      read(*,*)a,b,c
      if(b*b-4*a*c) 5,4,10
 5    rp=-b/2*a
      ip=sqrt(abs(b*b-4*a*c))/2*a
      write(*,*)'The imaginary roots are: ',rp,'+i',ip,'and',rp,'-i',ip
      goto 100  ! this goto statement transfers the control to the stop statement.
                ! otherwise control goes to immediate next statement.
 4    r1=-b/2*a
      write(*,*)'Roots are real and equal which are:',r1,r1
      goto 100
 10   r1=(-b+sqrt(b*b-4*a*c))/2*a
      r2=(-b-sqrt(b*b-4*a*c))/2*a
      write(*,*)'Roots are real and unequal and which are:',r1,r2
100   stop
      end
```

Output:

Run1:

```
Enter the values of a,b and c:
1   -10     25
Roots are real and equal and which are: 5. 5.
```

Run2:

```
Enter the values of a,b and c:
1   2   -35
 Roots are real and unequal and which are: 5. -7.
```

Run3:

```
Enter the values of a,b and c:
1    5    36
The imaginary roots are: -2.5 +i 5.454356 and -2.5 –i 5.454356
```

Question: Write a program to evaluate the following function using the concept of arithmetic if statement.

$f(y) = 3y^2-\cos(5y)$ if $y<5$

$=100$ if $y=0$

$=3y^2+\sin(5y)$ if $y>5$

if…then…else statement

This statement is more easer than the logical statement. It is similar to the **if..else** statement of C. The general form of this statement is:

```
if(test condition)  then
        statement 1
        statement 2
        ……
        ……
        statement n
else
        statement 1
        statement 2
        ……
        ……
        statement n
endif
```

The concept of control flow in this statement is similar to the if…else statement in C. There are two branches in the general form. If the **test condition** result in **true** then the if block statements will be executed i,e **statement 1** through **statement n** inside if block. Similarly, **statenent 1** through **statement n** inside else block will be executed if the **test condition** gives **false** result. **endif** statement closes the block of **if…then…else** statement. **else** block is optional in this structure. If this block is eliminated from this structure seems as

```
if(test condition)  then
        statement 1
        statement 2
        ……
        ……
        statement n
endif
```

It resembles to the simple **if** statement of C.

Example 20: A program to illustrate the simple if statement. This example, reads the age of a person, calculates his bonus and displays it. There is a rule to increase his bonus by 10% if his age is greater than 60.

```
real age, bonus
write(*,*) 'Enter values of age:'
read(*,*),age
write(*,*) 'Enter values of bonus:'
read(*,*) bonus
if(age.gt.60)then
        bonus=bonus+bonus*1.0/10.0
endif
write(*,*) 'Total bonus is:',bonus
end
```

Question: Write a program to read an integer and divide it by 10 and display if it is greater than 0.

Question: Write a program to read the age of a person and display message whether the he/she has voting right or not. Generally, a person greater than 18 years old has the right.

Example 21: Write a program to find the higher of two integers entered by the user. This example illustrates the concept of if…then…else statement.

```
write(*,*),'Enter two integers:'
read(*,*) i,j   ! why variables i and j are not declared.?
        if(i.gt.j) then
                write(*,*) ' greater is',i
        else
                write(*,*) ' greater is',j
        endif
end
```

Question: Write a program to read a number and display the message whether the number is odd or even.

Example 22: A program to calculate the area of a triangle when the length of three sides is given. This program illustrates the concept of if…else, relational operators and logical operators.

```
real a,b,c,s,area
write(*,*),'Enter the sides of the triange:'
read(*,*),a,b,c
if((a+b).gt.c .and. (b+c).gt. a .and. (a+c).gt.b) then
        s=(a+b+c)/2
        area=sqrt(s*(s-a)*(s-b)*(s-c))
        write(*,*),'The area of triangle is:',area
else
        write(*,*),'Invalid sides of the triangle.'
endif
end
```

Example 23: Write a program to check whether a year entered by a user is leap year or not. This example illustrates the concept of nested if…then…else statement.

```
integer year
write(*,*)'Enter a year:'
read(*,*) year
if(mod(year,4).eq.0)then
        if(mod(year,100).eq.0)then
                if(mod(year,400).eq.0)then
                        write(*,*)'the year',year,'is a leap year.'
                else
                        write(*,*)'the year',year,'is not a leap year.'
                endif
        else
                write(*,*)'The year',year,'is a leap year.'
        endif
 else
        write(*,*)'Entered year',year,'is not a lear year.'
```

```
    endif
    end
```

output:

Run1:

Enter a year:
2000
the year 2000 is a leap year.

Run2:

Enter a year:
1900
the year 1900 is not a leap year.

Run3:

Enter a year:
1996
The year 1996 is a leap year.

Question: Rewrite the above program by swapping the body of outermost if…then…else statement i,e. body of if in else and vice versa.

Question: Write a program to find the highest of three numbers entered by the user using nested if…then…else statement.

Question: Write a program to read a positive integer and display its square if it is even else its square root.

Question: Write three points of a rectangular coordinate system. Check whether joining these points can form a triangle. If yes, compute and display the perimeter and area of the triangle else display appropriate message.

else…if…then structure

This statement is multi-way decision-making statement. It is similar to the else…if structure of C language. Its general form is

```
if(test condition 1) then
        statement block 1
        ……...
        else if( test condition 2) then
                statement block 2
                ……...
                ……...
                else if( test condition n) then
                        statement block n
                        ……...
                        else
                                else block statement
                                …….. .
endif
```

In this structure, if the **test condition 1** gives the true result, then the **statement block 1** will be executed and **statement block 2** will be executed if **test condition 2** produce true.

Similarly, control will be transferred to **statement block n** if the **test condition n** gives true result. If no test condition gives true result, then else block statement will be executed. This concept is illustrated in the following example.

Example 24: Write a program to find the highest of three numbers entered by the user using else…if…then statement and logical expression.

```
real a,b,c,highest
write(*,*)'Enter three different real numbers:'
read(*,*)a,b,c
if(a .gt. b .and. a .gt. c)then
    highest=a
else if(b .gt. a .and. b .gt. c)then
    highest=b
else
    highest=c
endif
write(*,*)'Highest among three is', highest
end
```

Output:

Enter three different real numbers:
20.4 200.6 10.3
Highest among three is 200.6

Question: Write a program to read a day number and display whether it is Sunday, Monday, Tuesday, Wednesday, Thursday, Friday and Saturday.

Question: Write a program to read a month number and display the name of corresponding month.

Question: Write a program to read marks of a student and display his grade A if marks greater than or equal to 80, B if marks greater than or equal to 65 and C if marks greater than or equal to 40 else F for fail.

Question: Write a program, which computes the weekly commission earned by a salesman as per the sales amount based on the following criteria.

Sales amount (in rupees)	**Commission (In Rupees)**
≤10000	2000
>10000 to 20000	2000+22% on extra (above Rs. 10000)
>20000 to 30000	4200+25% on extra (above Rs. 20000)
>30000	6700+27% on extra (above Rs. 30000)

Question: Write a program to read a coordinate point of a rectangular coordinate system and display the appropriate message whether the point lies at center or on any axis or on any quadrants.

Question: Write a program to evaluate the following function f(x) given by

=0 if x≤0
= x(x-10)(x-15) if 0<x≤10
= (x-10)(x-15)(x-20) if 10<x≤15
F(x)= (x-15)(x-20)(x-30) if 15<x≤20
=(x-20)(x-30)(x-40) if 20<x≤30
= 0 for all other cases

do loops

This statement is used for executing a block of code repeatedly. This is counter controlled loop mechanism. It is considered as the most effective tool in ForTran. The working mechanism of this loop is similar to **for** loop of C. The general form of **do** loop structure is

```
        do n I = start, stop, update
                ……..
                ……..
n       continue
```

Where, **n** is the valid statement number of the last statement of the body of the loop. Generally **continue** is used as the last statement. **I** is an integer variable. It is used to control loop structure. So, it is also called loop control or running variable. **start** is the starting value of the **I** from which the loop execution begins. **stop** indicates the stopping value of the **I**. It indicates loop structure must not go beyond the stop. **update** increments or decrements the **I** for the execution of next loop round. **update** is optional. If it is not given, default value for increment is 1. Although some ForTran77 compilers accept real, the basic type of **start, stop** and **update** is integer. These can be constants, expressions or variables. **Continue** statement indicates the end of the do loop structure. It must have a label i,e statement number. It is a dummy statement. It is only a placeholder. The portion between **do** and **continue** statement is called the body of do loop. For example, to print from 10 to 100 in step of 2 general form becomes

```
        do n I = 10, 100, 2
                ……..
                ……..
n       continue
```

but, to print in reverse order it becomes

```
        do n I = 100, 10, -2
                ……..
                ……..
n       continue
```

The application of this loop is illustrated in the following examples.

Example 25: Write a program to read two integers n1 and n2 (n1<n2) and display the integers in the range inclusive.

```
integer n1,n2,k
write(*,*)'Enter the value of n1 and n2:'
        read(*,*) n1,n2
        do 1 k=n1,n2,1
                write(*,*)k
  1     continue
        end
```

Output:

```
Enter the value of n1 and n2:
10      14
```

```
10
11
12
13
14
```

Example 26: Write a program to read two integers n2 and n1 (n2>n1) and display the integers in the range from n2 to n1 inclusive.

```
      integer n2,n1,k
      write(*,*)'Enter the value of n1 and n2:'
      read(*,*) n2,n1
      do 1 k=n2,n1,-1
            write(*,*)k
1     continue
      end
```

Output:

```
Enter the value of n1 and n2:
1000        996
1000
999
998
997
996
```

Example 27: Write a program to read four integers greater than zero and displays the highest and lowest among them.

```
      integer n,h,l
      h=0
      l=0
      do 1 k=1,4,1                    !why k is not declared for its type here
            write(*,*)'Enter number a number:'
            read(*,*)n
            if(n.gt.h)then
                  h=n
            endif
            if(n.lt.l)then
                  l=n
            endif
1     continue
      write(*,*)'Highest',h,'Lowest',l
      end
```

Output:

```
Enter number a number: 1000
Enter number a number: -2000
Enter number a number: 3786
Enter number a number: 22
Highest 3786 Lowest –2000
```

Question: Modify this program to find the highest, lowest and average of n numbers i,e, read n from the user instead of using constant value.

Question: Write a program to find the sum of even integers from 1 to n. Where n is a positive integer greater than 1 entered by the user.

Example 28: Write a program to read x as a real number and n as an integer and display the value of $x^n/n!$. Calculate x^n and n! using do loops.

```
      integer n,f
      real x,p
      f=1
      p=1
      write(*,*)'Enter values of x and n:'
      read(*,*) x,n
      do 1 k=1,n,1
            p=p*x
1     continue
      do 2 k=1,n,1
            f=f*k
2     continue
      write(*,*)x,'^',n,'/',n,'!:',p/f
      end
```

Output:

```
Enter values of x and n: 2     4
 2. ^ 4 / 4 !: 0.6666667
```

Example 29: Write a program to evaluate the exponential series e^x $=1+x+x^2/2!+x^3/3!+x^4/4! +x^5/5!$……upto n terms.

```
      write(*,*)'Enter values of x and number of terms:'
      read(*,*) x,n
      term=1
      do 1 k=1,n,1  !why x,k,n,term,sum are not declared?
            term=term*x/k
            sum=sum+term
            write(*,*)'term',k,':',term
1     continue
      write(*,*)'Sum of terms is:',sum
      end
```

Output:

```
 5  5
 term 1 : 5.
 term 2 : 12.5
 term 3 : 20.833334
 term 4 : 26.041668
 term 5 : 26.041668
 Sum of terms is: 90.41667
```

Nested do loops

The concept of nested do loop is similar to the concept of nested for loop in C. In the nested structure of do loops, each loop must have its own corresponding continue statement. Its general form is as:

do n **I** = start, stop, update

……..

……..

do n1 **I1** = start1, stop1, update1

```
                                        ……..
                                        ……..
        n1              continue
          n       continue
```

Example 30: Write a program to find the terms display them. Display sum of terms also. This example illustrates the application of nested do loop.
s=1+(1+2)+(1+2+3)+(1+2+3+4)+……(1+2+3+4+…….+n)

```
        integer sum,term,n
        write(*,*)'Enter number terms:'
        read(*,*)n
        sum=0
        do 1 k=1,n,1
            term=0
                do 2 j=1,k,1
                    term=term+j
2               continue
                 sum=sum+term
            write(*,*) 'term',k,term
1       continue
        write(*,*) 'Sum of all terms:',sum
        end
```

Output:

```
Enter number terms:
4
 term 1   1
 term 2   3
 term 3   6
 term 4   10
 Sum of all terms:          20
```

Example 31: Write a program to ask the user to enter to positive numbers f and s (f>1,s>1 and f<s). Display the prime numbers between the numbers inclusive. (Modify this program to count them and display the count also.)

```
        integer f,s,k,check
        write(*,*)'Enter number first and second number:'
        read(*,*)f,s
        do 1 k=f,s,1
            check=1
            do 2 j=2,k/2,1
                if(mod(k,j).eq.0)then
                    check=0
                endif
2           continue
            if(check.eq.1)then
                    write(*,*) k,'is prime.'
            endif
1       continue
        end
```

Output:

Enter number first and second number:
20 40
23 is prime.
29 is prime.
31 is prime.
37 is prime.

Question: Write a program to read two years year1 and year2 (year1<year2) and display the leap years between them inclusive.

Question: Write a program to read an integer greater than 10 and display square root, square, cube and cube root of numbers from 1 to n inclusive.

Question: Write a program to evaluate the expression **$s=ut+(at^2)/2$** from **t**=0 to 10 in the interval of 1. Read value of **u** and **a** from the user. Make it more general by reading start and end values of time.

Question: Write a program to print first n terms of the Fibonacci sequence.

Question: Write a program to read n data from the user until the user enters zero and display their range.

Question: Write a program that read in two integers i and j print out the all integer between them in reverse order. Do not include i and j in the output. Display the count of those numbers, which are both multiple of 5 and 7 in the range

Question: Write a program to read an integer n (|n|>100) find the sum from n to 0 if it is negative else count the twin primes between 10 to n.

Question: Write a program to print the sum of terms generated by the given expression. Also display the sum of the terms.

$$\sum_{i=1}^{n} \left((-1)^{(i+1)} \right) / i!$$

Question: write a program to evaluate the following expression. Read the value of x, y, n from the user and display the result.

$$f(x,y) = \sum_{i=1}^{n} \left(x^i + y^i \right)$$

Questions: Write a program to evaluate the following series up to n terms

a. $s=1+2+3+4+5+\ldots\ldots\ldots\ldots n$
b. $s=2+4+6+8+10+12+14\ldots\ldots\ldots\ldots 2n$
c. $s=1+3+5+7+9+11+13\ldots\ldots\ldots\ldots 2n-1$
d. $s=1^2+2^2+3^2+4^2+5^2+6^2+\ldots\ldots\ldots\ldots n^2$
e. $s=1^3+2^3+3^3+4^3+5^3+6^3+\ldots\ldots\ldots\ldots n^3$
f. $s=1+1/2+1/3+1/4+1/5+\ldots\ldots\ldots\ldots 1/n$
g. $s=1+1/2^2+1/3^2+1/4^2+1/5^2+\ldots\ldots\ldots\ldots 1/n^2$
h. $s=1+1/3+1/5+1/7+1/9+1/11+1/13\ldots\ldots 1/2n-1$
i. $s=1+(1+2)+(1+2+3)+(1+2+3+4)+\ldots\ldots(1+2+3+4+\ldots\ldots+n)$
j. $\pi=4(1-1/3+1/5-1/7+1/9-1/11\ldots\ldots\ldots\ldots 1/(2n-1))$
k. $e =1+1/1!+1/2!+1/3!+1/4! +1/5!\ldots\ldots 1/n!,\ n=0,1,2,3\ldots\ldots$
l. $e^{-1} =1-1/1!+1/2!-1/3!+1/4! -1/5!\ldots\ldots. (-1)^{n+1}/(n-1)!,\ n=1,2,3\ldots.$
m. $S= 1+2+3+4+5+\ldots..+n-2+n-1+n+n-1+n-2+\ldots..+5+4+3+2+1$
n. $S=n+n-1+n-2+\ldots..+ 5+4+3+2+1+2+3+4+5+\ldots..n-2+n-1+n$

o. $f(x)=1+x+x^2+x^3+x^4+x^5+\ldots\ldots\ldots\ldots+x^n$
p. $e^x=1+x+x^2/2!+x^3/3!+x^4/4!+x^5/5!\ldots\ldots x^n/n!$, n=0,1,2,3…….
q. $e^{-x}=1-x+x^2/2!-x^3/3!+x^4/4!-x^5/5!\ldots\ldots (-1)^{n+1}x^{(n-1)}/(n-1)!$, n=1,2,3….
r. $\sin(x)= x-x^3/3!+x^5/5!-x^7/7!\ldots\ldots\ldots (-1)^{(n-1)} x^{(2n-1)}/(2n-1)!$, n=1,2,3,4….
s. $\cos(x)= 1-x^2/2!+x^4/4!-x^6/6!+x^8/8!-\ldots\ldots\ldots (-1)^{(n)} x^{(2n)}/(2n)!$, n=1,2,3,4….
t. $\sinh(x) = (e^x- e^{-x})/2$
u. $\cosh(x) = (e^x+ e^{-x})/2$
v. $\tanh(x)= (e^x- e^{-x})/(e^x+e^{-x})$

11.7 Arrays

One dimensional arrays

An array is a sequence of consecutive memory location. We have learned different array processing in C. The theoretical concept is same to that. Here, we study how to manipulate arrays using ForTran syntax. Array indexing in ForTran starts from 1 where as in it starts from 0 in C. The subscript must be positive integer and given with in parenthesis. If there are more than one subscripts, they must be separated by commas. The **dimension** statement is there to declare arrays in ForTran. Its general form is

dimension array_name (size of array)

for example, **dimension a(5)** is an array declaration of size 5, name **a.** Subscript varies from 1 to 5. Instead of **dimension,** integer, real, character, double precision, complex and local statements can also be used to declare arrays.

Example 32: This program shows the technique of reading and printing members of a one dimensional array.

```
      dimension a(5)
      do 1 k=1, 5, 1
            read(*,*) a(k)
1     continue
      do 2 k=1, 5, 1
            write(*,*) a(k)
2     continue
      end
```

Here do loop is written explicitly for reading and writing array members. This can be done more shortly using implied do loop. The difference between the normal and implied do loop structure is shown in following table.

Comparison between normal and implied do loop structure.		
Normal do loop structure	**Implied do loop structure**	**purpose**
do 1 k=1, 5, 1 read(*,*) a(k) 1 continue	read(*,*) (a(i),i=1,5)	To read data for array members a(1), a(2), a(3), a(4), and a(5)
do 2 k=1, 5, 1 write(*,*) a(k) 2 continue	write(*,*) (a(i),i=1,5)	To display content of array members a(1), a(2), a(3), a(4), and a(5)

The application of one-dimensional array processing is illustrated in the following examples.

Example 33: Write a program to declare an array of size 5. Read and display array members using implied do loops.

```
dimension a(5)
write(*,*) 'Enter array members:'
read(*,*) (a(i),i=1,5)
write(*,*) 'The read array members are:'
write(*,*) (a(i),i=1,5)
end
```

Output:

```
Enter array members:
10.3    30      -40.6     -100       20.2
 The read array members are:
 10.3    30.    -40.6    -100.    20.2
```

Example 34: Write a program to read members of array of size 5 of type real and display those members which are exactly divisible by 3 but not by 7.

```
      real a(5)
      write(*,*) 'Enter array members:'
      read(*,*) (a(i),i=1,5)
      do 1 i=1, 5, 1
            if(mod(a(i),3.0).eq.0.and.mod(a(i),7.0).ne.0)then
                  write(*,*)'Required number is:',a(i)
            endif
1     continue
      end
```

Output:

```
Enter array members:
15  21  42   18   63
 Required number is: 15.
 Required number is: 18.
```

Questions: Write a program

a. to find sum of all members of the array.
b. to find the sum of odd and even members separately.
c. to count odd, even, prime and Armstrong members.
d. to display the members which are odd and prime.
e. to display the members which are Armstrong and even
f. to display the members which are Armstrong and odd
g. to display the members which are twin primes.
h. to count the palindrome members
i. to display the highest and lowest member as well their index,
j. to display sum and difference of highest and lowest members.
k. to reverse the array.

l. to calculate the sum, difference, product and quotient of members at array position 2 and 5. For this array size must be greater or equal to 5.
m. to swap members at array position 2 and 5. For this array size must be greater or equal to 5.
n. to modify the value of array position at 5. For this array size must be greater or equal to 5.
o. to find the sum of digits of the array members and assign the sum in the respective position of the array. For example, 123 is a member in third position then find 1+2+3 and store 6 at third position.
p. to reverse the array members and assign the result at the respective position. For instance, 123 is a member in third position then, reverse i,e, 321 and store it at third position.
q. to assign 1 to the respective position if the array member is odd and 2 if it is even.
r. to raise the power of each member by 3 and store at the respective position

Note: select the type of array members as the nature of problem. Think about two-dimensional arrays for these problems.

Example 35: Write a program to declare three arrays of type real. Read the members of first two arrays and sum of member of corresponding position and assign the result to the corresponding position of the third array and display the members of the third array.

```
      real a(50),b(50),c(50)
      write(*,*) 'Enter size of arrays:'
      read(*,*)n    !why n is not declared
      write(*,*)'Enter members of array a:'
      read(*,*)(a(i),i=1,n,1)     !why n is not declared
      write(*,*)'Enter members of array b:'
      read(*,*)(b(i),i=1,n,1)
      do 1 i=1,n,1
                  c(i)=a(i)+b(i)
1     continue
      write(*,*)'members of array c after addition:'
      write(*,*)(c(i),i=1,n,1)
      end
```

Output:

```
Enter size of arrays:
5
 Enter members of array a:
2.3       4.5       5.6      5       33
 Enter members of array b:
2.4       -4.5       4.6      2       7.5
members of array c after addition:
 4.7       0.        10.2    7.      40.5
```

Questions: Write separate programs

a. To find sum of member of corresponding position and assign the result to the corresponding position of the second/first array and display the result.
b. To find difference of member of corresponding position and assign the result to the corresponding position of the second/first array and display the result.

c. To swap the member of corresponding position of two arrays. And display the members of both arrays.
d. To reverse the second array and add it to the first and assign the result in the first array and display the result i,e, do not use the third array to store the result.

Example 36: Write a program to declare an integer array of size 5, read the members of the array, sort them in ascending order using bubble sort algorithm and display the sorted array. Modify this program for descending order also.

```
      write(*,*)'Enter array members:'
      read(*,*)(marks(i),i=1,5,1)
      do 1 p=1,5-1,1
            do 2 c=1,5-p-1,1
                  if(marks(c).gt.marks(c+1))then
                        temp=marks(c)
                        marks(c)=marks(c+1)
                        marks(c+1)=temp
                  endif
2           continue
1     continue
      write(*,*)'sorted array is:'
      write(*,*)(marks(i),i=1,5,1)
      end
```

Output:

```
Enter array members:
-30   200  -500   1   1000
 sorted array is:
 -500   -30  1   200   1000
```

Example 37: Write a program along with algorithm and flowchart to compute the arithmetic mean (A.M.), geometric mean (G.M.) and harmonic mean (H.M.) of any numbers of given data values (x).

a. G.M.= $(x_1 x x_2 x x_3 x x_4 x \ldots\ldots\ldots x x_n)^{1/n}$

b. H.M.= $\dfrac{n}{1/x_1 + 1/x_2 + 1/x_3 + 1/x_4 + \ldots\ldots\ldots\ldots + 1/x_n}$

c. A.M.= $\sum_{i=1}^{n} x_i$

```
real x(50)
write(*,*) 'Enter size of array:'
read(*,*)n
write(*,*)'Enter members of array x:'
read(*,*)(x(i),i=1,n,1)
gmp=1
ams=0
hms=0
do 1 i=1,n,1
      gmp=gmp*x(i)
      hms=hms+(1/x(i))
      ams=ams+x(i)
1     continue
```

Question: Why variables (except array) used in this program are not declared?

```
gm=gmp**(1.0/n)
hm=n/hms
am=ams/n
write(*,*)'GM:',gm,'  HM:',hm,'  AM:',am
end
```

Output:

```
Enter size of array:
5
Enter members of array x:
1    2    3    4  5
GM: 2.6051712   HM: 2.1897807   AM: 3.
```

Question: Write separate programs to compute the median, range, variance and standard deviation. Note: Variance= $(\Sigma x^2)/n - ((\Sigma x^2)/n)^2$ and standard deviation is $\sqrt{}$(variance).

Two-dimensional Arrays

Study the following examples to understand the syntactic structure of two-dimensional arrays processing in ForTran77.

Example 38:This example simply shows how to read and write(display, print) members of two dimensional arrays. In this example implied do loops are used for reading and writing members row wise.

```
      integer x(3,3) ! two dimensional array declaration
      write(*,*)'Enter the members of the array:'
      do 1 i=1,3,1
            read(*,*)(x(i,j),j=1,3,1)   ! Reading members of two dimensional array row wise
1     continue
      write(*,*)'The members of the read array:'
      do 2 i=1,3,1
            write(*,*)(x(i,j),j=1,3,1) ! Displaying members of two dimensional array row wise
2     continue
      end
```

Output:

```
Enter the members of the array:
1    2    3
4    5    6
7    88   9
 The members of the read array:
 1    2    3
 4    5    6
 7    88   9
```

Review question: Write a program to compute the following expressions using library function.

a. $x=10t^2-t^5-2$
b. $y=sint+cos2t$
c. $z=e^{3t}+5$

Read the value of t from the user.

Example 39: Write a program to transpose a 3x3 matrix.

```
      real x(3,3),tem
      do 1 i=1,3,1
      read(*,*)(x(i,j),j=1,3,1)
1     continue
      do 2 i=1,3,1      ! i for accessing rows and j for accessing columns
            do 3 j=1,3,1
                  if(i.lt.j)then
                        temp=x(i,j)
                        x(i,j)=x(j,i)
                        x(j,i)=temp
                  endif
3           continue
2     continue
      write(*,*)'Transposed matrix is:'
      do 4 i=1,3,1
            write(*,*)(x(i,j),j=1,3,1)
4     continue
      end
```

Output:

```
Enter members of array x row wise:
1.5      2.5      3.5
2.3      2.4      2.5
-3.4    -3.9     -3.8
 Transposed matrix is:
 1.5      2.3     -3.4
 2.5      2.4     -3.9
 3.5      2.5     -3.8
```

Example 40: Write program to add and subtract two matrices of type real and display the result.

```
      integer x(10,10),y(10,10),s(10,10),d(10,10),r1,c1,r2,c2 !arrays and variable declaration
      write(*,*)'Enter row and column size of arrays x and y:'
      read(*,*)r1,c1,r2,c2
      if(r1.eq.r2 .and. c1.eq.c2)then   !checking valid condition for matrix addition and subtraction
            write(*,*)'Enter members of array x row wise:'
            do 1 i=1,r1,1
                  read(*,*)(x(i,j),j=1,c1,1) !reading members of array x
1           continue
            write(*,*)'Enter members of array x row wise:'
            do 2 i=1,r2,1
                  read(*,*)(y(i,j),j=1,c2,1) !reading members of array y
2           continue
            do 3 i=1,r1,1
                  do 4 j=1,c1,1
                        s(i,j)=x(i,j)+y(i,j) ! Adding matrices x and y and assigning result to matrix s
                        d(i,j)=x(i,j)-y(i,j) ! Subtracting matrix y from x and assigning the result to matrix d
4                 continue
3           continue
            write(*,*)'The members of array s:'
            do 5 i=1,r1,1
```

```
            write(*,*)(s(i,j),j=1,c1,1) ! printing elements of matrix s
5           continue
            write(*,*)'The members of array d:'
            do 6 i=1,r1,1
                write(*,*)(d(i,j),j=1,c1,1)  ! printing elements of matrix d
6           continue
    else
            write(*,*)'Matrix addition or subtraction is impossible.'
    endif
    end
```

Output:

```
Enter row and column size of arrays x and y:
3 3  3  3
 Enter members of array x row wise:
1  2  3
4  5  6
7  8  9
 Enter members of array x row wise:
10  20  30
40  50  60
70  80  90
 The members of array s:
 11 22 33
 44 55 66
 77 88 99
 The members of array d:
 -9 -18 -27
 -36 -45 -54
 -63 -72 -81
```

Questions: Write a program

a. to compute the product of two matrices of size mxn and pxq.
b. to find the sum of all the members of a two dimensional array.
c. to calculate the sum of diagonal elements of a square matrix.
d. to find the sum of individual rows of a two-dimensional array and assign them to a one-dimensional array and display the content of one dimensional array.
e. to rise each member of the array by 4 and assign the result in the corresponding position.
f. to reads marks of 6 subjects of 40 students, compute and display the percentage obtained by each student, the top score and the count of successful student if the full and pass marks of each subjects are 100 and 40 for each subject respectively.
g. to generate a matrix of mxn whose elements are generated by the expression $m_{ij}=x^{(i+j)}$. Read m,n and x from the user.
h. to find the range of elements of a two dimensional array.
i. to set -1 to the members i<j, 0 to the members i=j else 1 as the members of a two dimensional array. Where i and j are indices of rows and columns.
j. to compute sum of all positive and all negative elements of a one/two dimensional array separately.

k. to compute R=X*Y+Z^T where R,X,Y and Z are matrices of valid order and Z^T is the Transpose of Z.

l. Write a program to calculate the cost of cost of operating an electrical device using formula C=(WTK)/1000. Where W is number of Watts, T is time in hours and K is the Cost(in rupees) pre kilowatt hours. Store the value of C in a three dimensional array when W varies from 100 watts to 1000 watts in steps of 100, T varies from 1 hours to 10 hours in step of 1 and K varies from Rs. 1.0 to 10.0 in steps of 1 rupee. Display the content of array.

Example 41: Write a program to print the following pattern of numbers. Ask the user to enter number of rows of the pattern.

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

```
      write(*,*)'Enter value of rows: '
      read(*,*)i
      do 1 k=1,i,1
            write(*,*)(j,j=1,k) ! using implied do loop to print value of j
   1  continue
      end
```

Example 42: write a program to print the following pattern of numbers. Ask the user to enter number of rows of the pattern.

```
5 4 3 2 1
4 3 2 1
3 2 1
2 1
1
```

```
      write(*,*)'Enter value of rows: '
      read(*,*)i
      do 1 k=i,1,-1
                  write(*,*)(j,j=k,1,-1) ! using implied do loop to print value of j
   1   continue
      end
```

References:

1. Ram Kumar, PROGRAMMING WITH ForTran 77, Tata Mc Graw Hill, New Delhi, 1996
2. C. Xavier, FORTRAN 77 and NUMERICAL METHODS, NEW AGE INTERNATIONAL PUBLISHERS, 1994